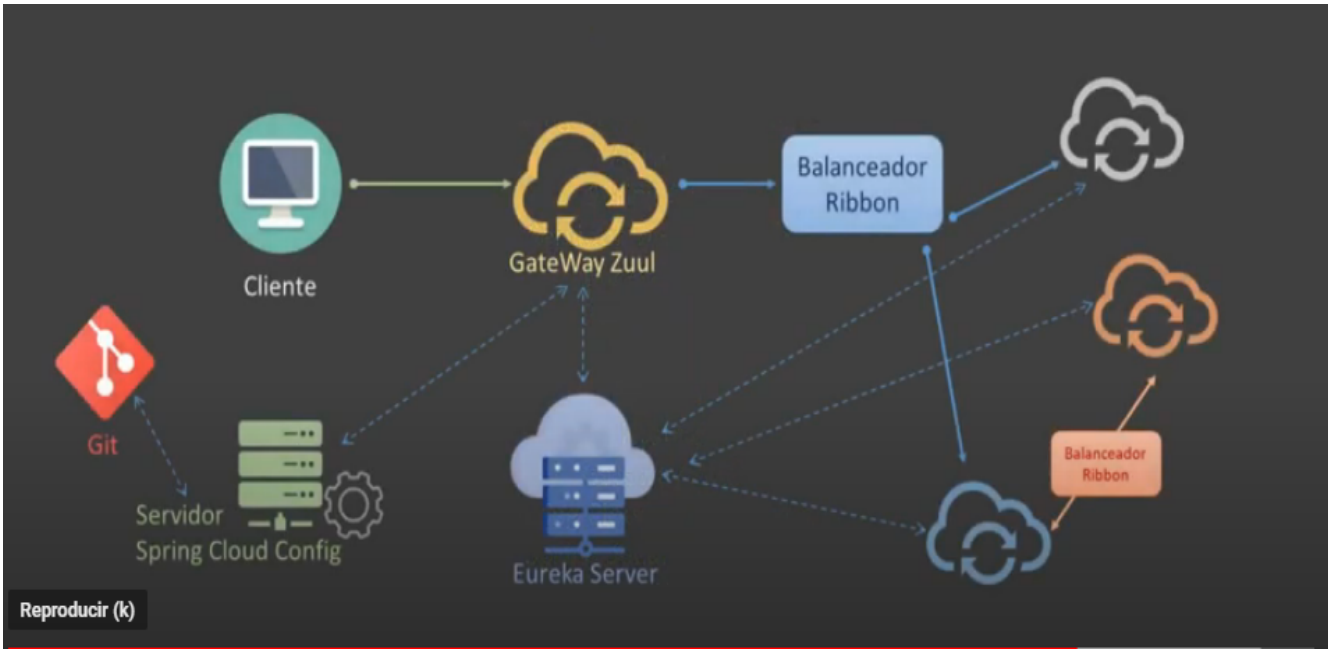
Ecosistema

La estructura inicial del ecosistema es la siguiente



Cliente :Front end en Angular PrimeFaces
Zuul: Es la puerta de entrada al ecosistema ,se puede usar spring cloud gateway
Ribbon: Es el balanceador de carga.
Spring Cloud Config: servidor de configuración de las propiedad de los microservicios
Eureka: Servidor de microservicios.

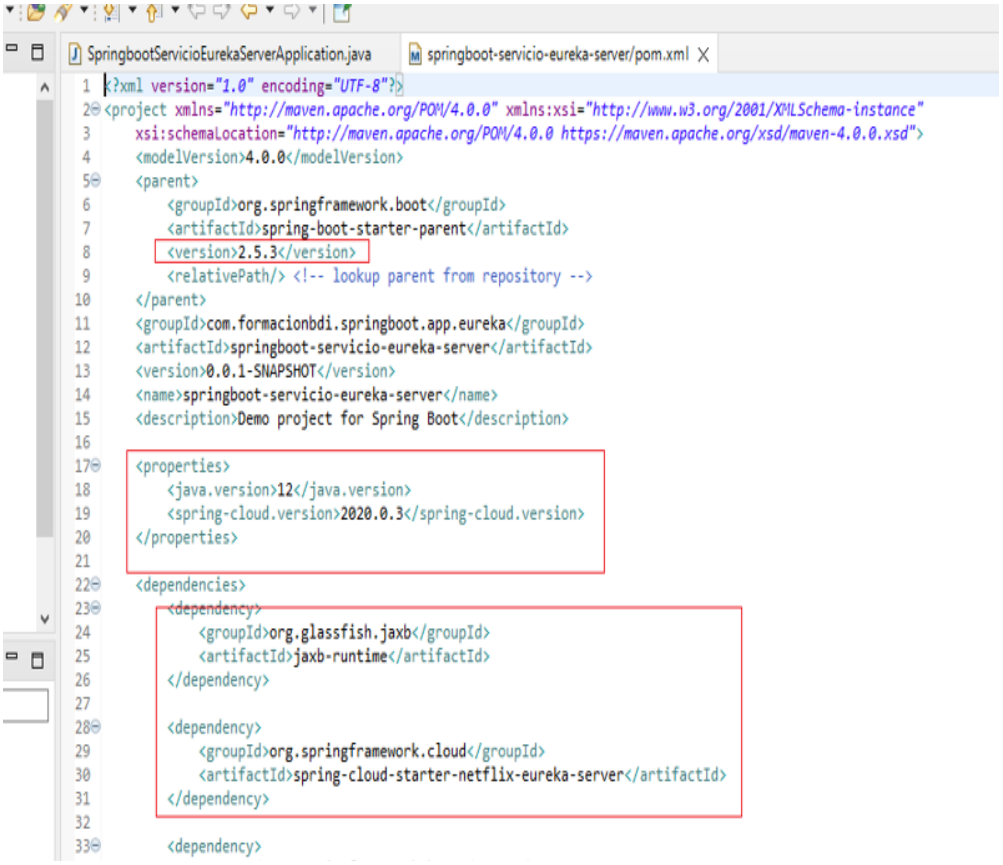
Configuración Eureka

SpringbootServicioEurekaServerApplication.java

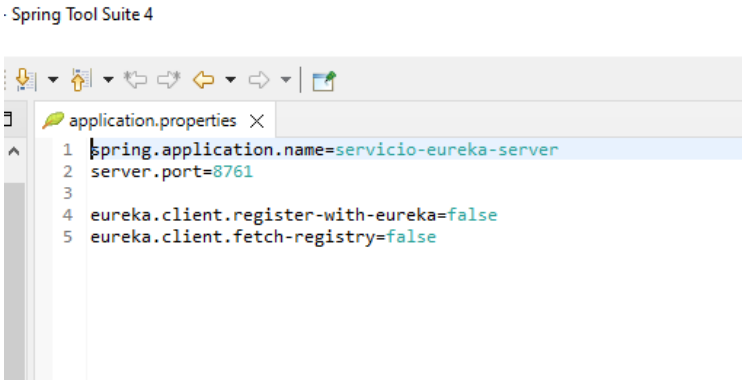
SpringbootServicioEurekaServerApplication.java

```
1  package com.formacionbdi.springboot.app.eureka;
2
3  import org.springframework.boot.SpringApplication;
6
7  @EnableEurekaServer
8  @SpringBootApplication
9  public class SpringbootServicioEurekaServerApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(SpringbootServicioEurekaServerApplication.class, args);
13     }
14
15 }
16
```

pom.xml



applications.properties



Para que los microservicios se registren en eureka como clientes con puertos dinámicos se deben configurar de la siguiente manera.
El ejemplo es del microServicio producto.

'productos/SpringbootServicioProductosApplication.java - Spring Tool Suite 4

SpringbootServicioProductosApplication.java | springboot-servicio-productos/pom.xml | application.properties

```java
package com.formacionbdi.springboot.app.productos;

import org.springframework.boot.SpringApplication;

@EnableEurekaClient
@SpringBootApplication
public class SpringbootServicioProductosApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootServicioProductosApplication.class, args);
    }

}
```

ngbootServicioProductosApplication.java | springboot-servicio-productos/pom.xml | application.properties

```xml
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
</dependencies>
```

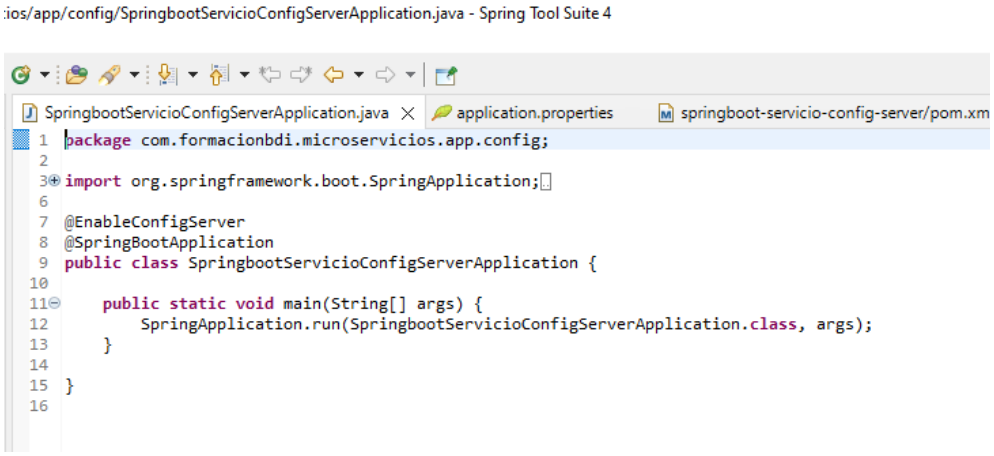
Tool Suite 4

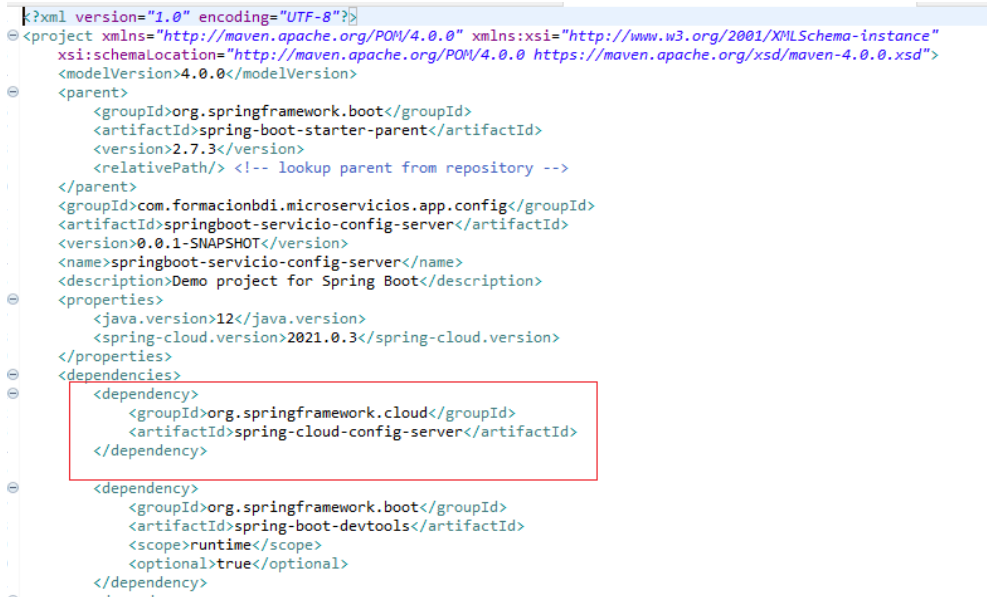
SpringbootServicioProductosApplication.java | springboot-servicio-productos/pom.xml | application.properties

```
spring.application.name=servicio-productos
#Para tener diferentes instancias en puertos aleatoria en Eureka
server.port=${PORT:0}
eureka.instance.instance-id=${spring.application.name}:${spring.application.instance_id:${random.value}}
#Ubicacion de Eureka
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

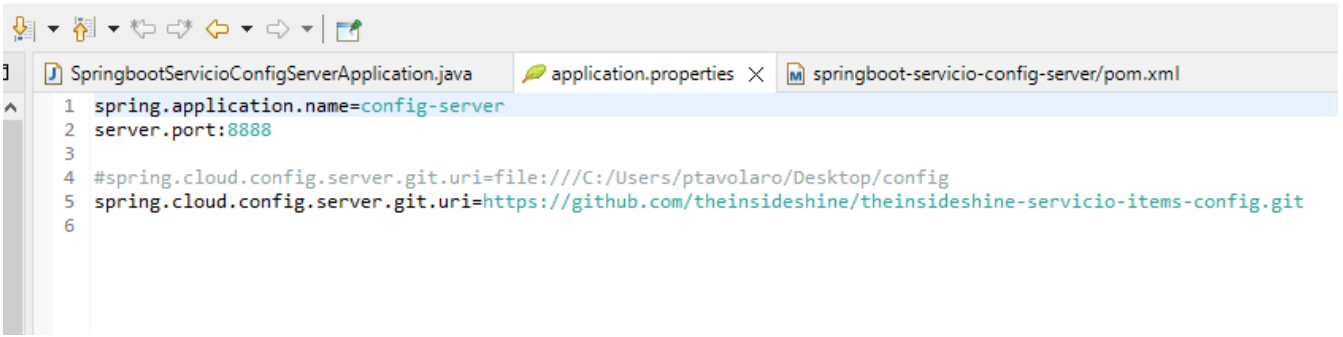
Spring Cloud Config

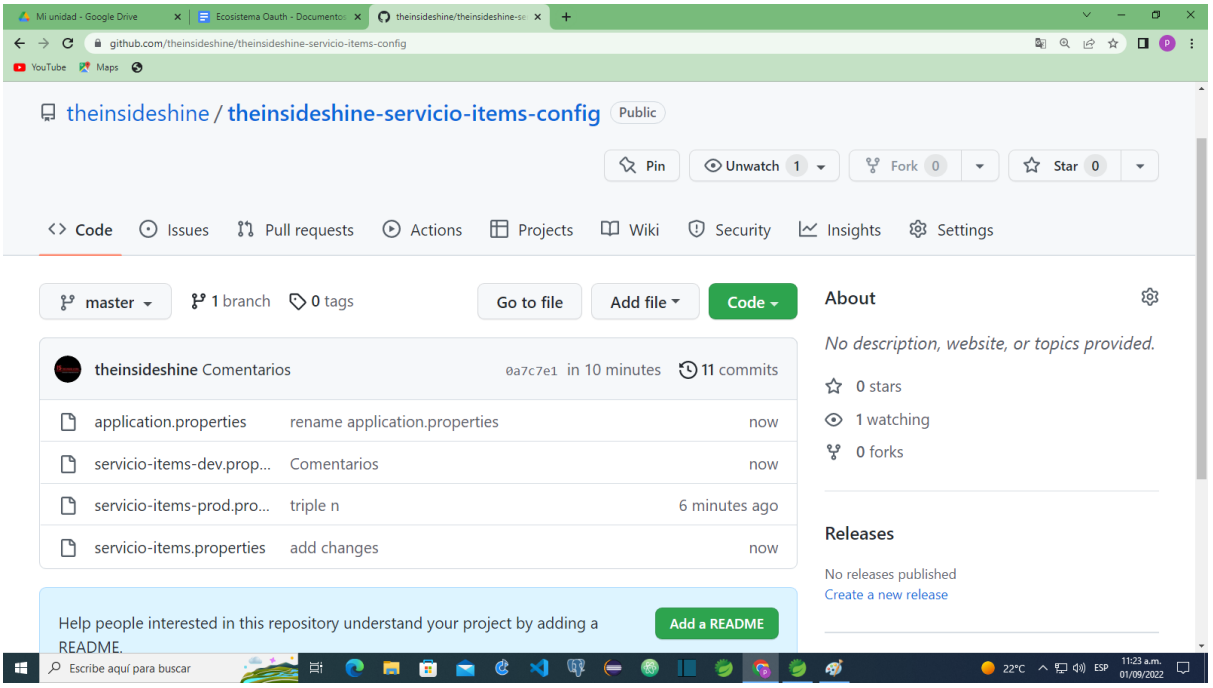
Las configuraciones como variables de entorno, pueden almacenarse en un servidor de configuración que está conectado a un repositorio, unos de los principales objetivos es poder cambiar la configuración sin reiniciar los microservicios.





Spring Tool Suite 4



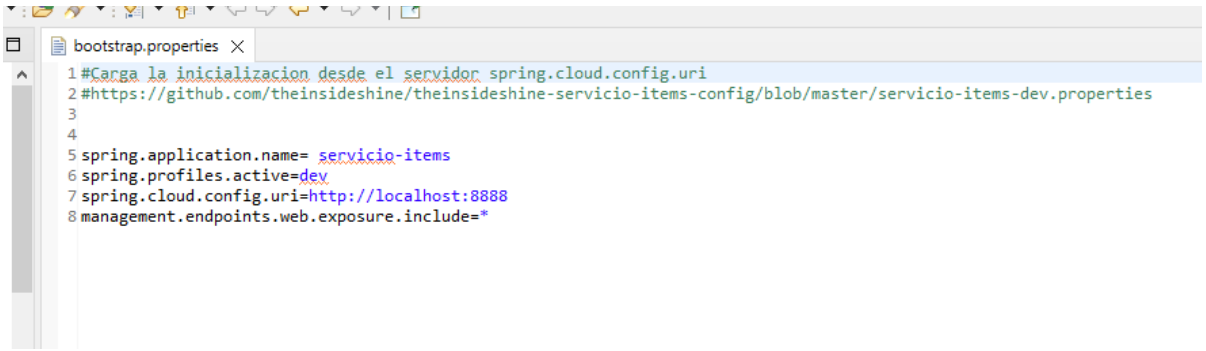


para actualizar la configuración acceder a la siguiente ruta:

POST localhost:port{Microservicio}/actuator/refresh

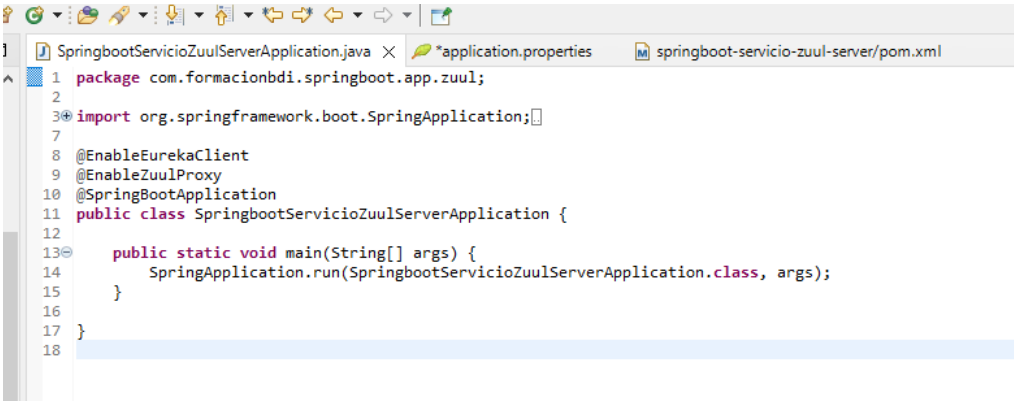
Para activar la configuración en cada microservicio se puede utilizar el archivo bootstrap.properties , donde localhost:8888 es el endpoint de servidor de configuración

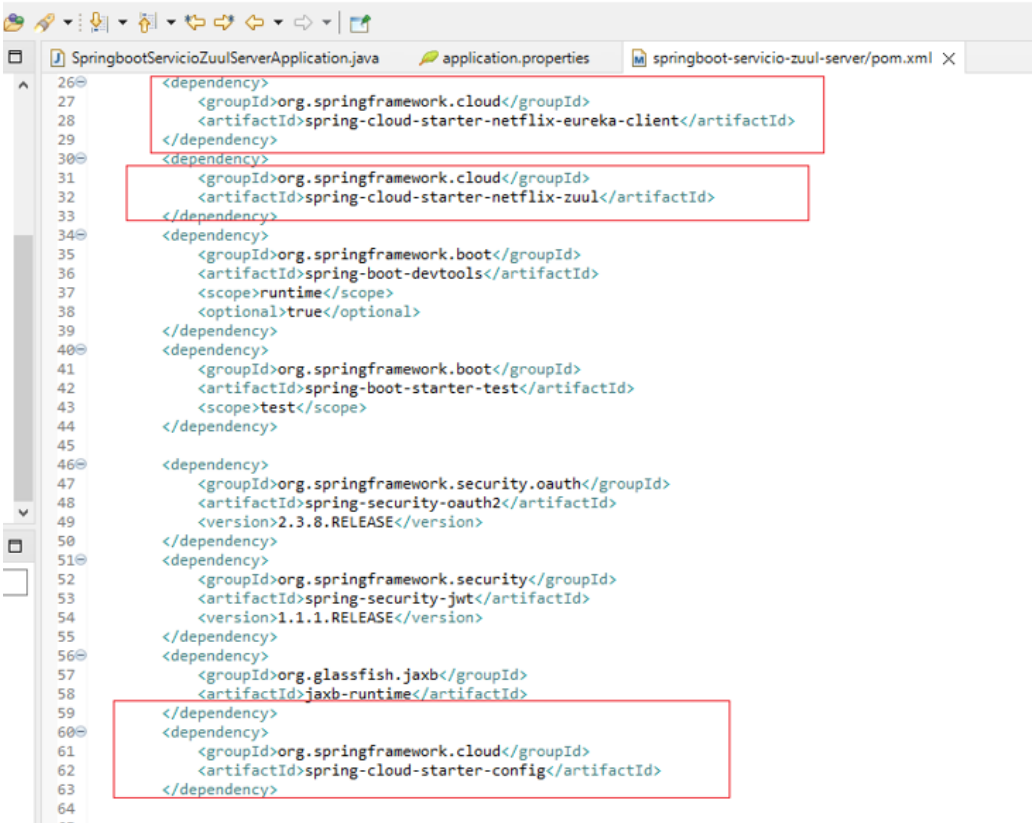
El ejemplo es del microServicio items.


bootstrap.properties
#Carga la inicializacion desde el servidor spring.cloud.config.uri
#https://github.com/theinsideshine/theinsideshine-servicio-items-config/blob/master/servicio-items-dev.properties
spring.application.name= servicio-items
spring.profiles.active=dev
spring.cloud.config.uri=http://localhost:8888
management.endpoints.web.exposure.include=*

GateWay: zuul ,desarrollado por netflix, está en modo mantenimiento.

Deben configurarse los nombres de todos los microservicios ,los timeout de los intentos por fallas.





pring Tool Suite 4

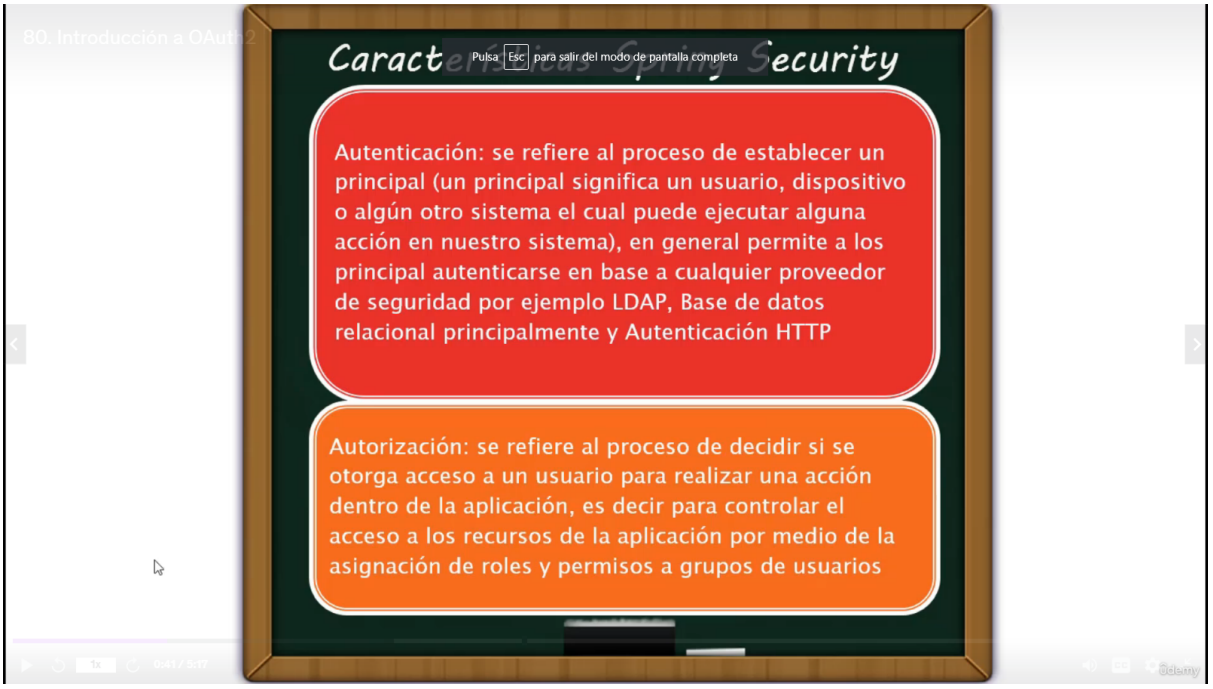
SpringbootServicioZuulServerApplication.java | application.properties | springboot-servicio-zuul-server/pom.xml

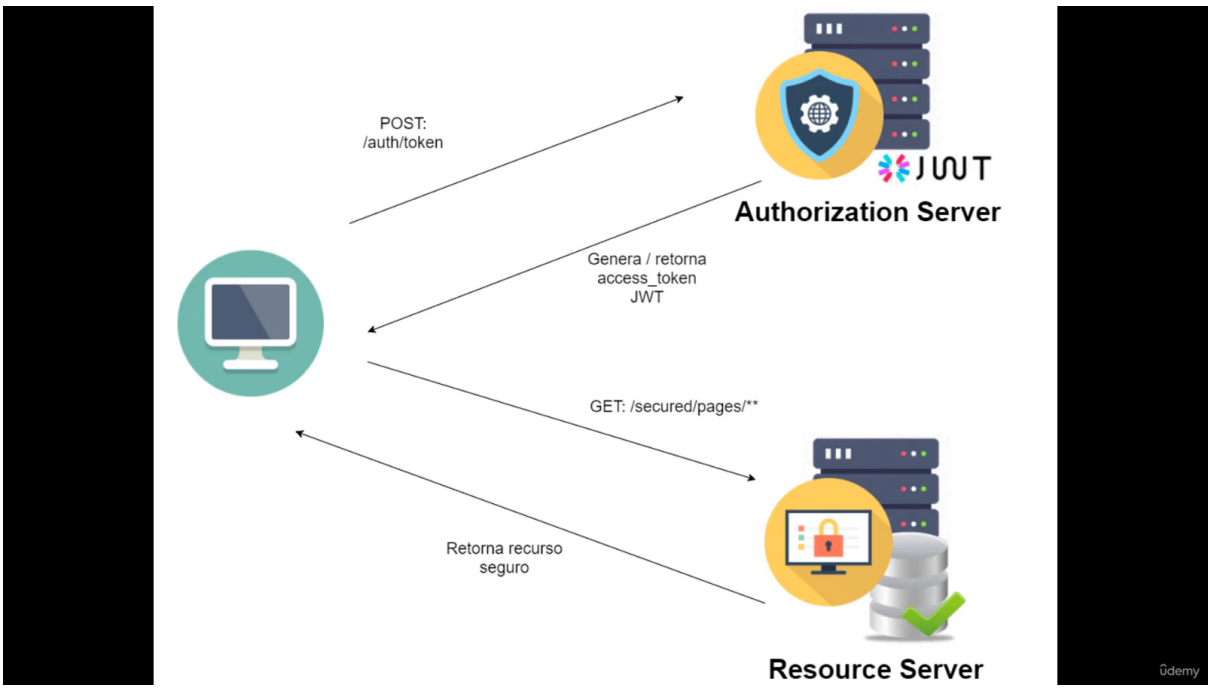
```
spring.application.name=servicio-zuul-server
server.port=8090

#servidor Eureka
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/

#microServicios

zuul.routes.productos.service-id=servicio-productos
zuul.routes.productos.path=/api/productos/**

zuul.routes.items.service-id=servicio-items
zuul.routes.items.path=/api/items/**

zuul.routes.usuarios.service-id=servicio-usuarios
zuul.routes.usuarios.path=/api/usuarios/**

zuul.routes.security.service-id=servicio-oauth
zuul.routes.security.path=/api/security/**
zuul.routes.security.sensitive-headers=Cookie,Set-Cookie


#timeOuts para sistema de fallos
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds: 20000
ribbon.ConnectTimeout: 3000
ribbon.ReadTimeout: 10000
```

Spring security

las dos partes principales de estas separadas

MicroServicio Oauth: autenticacion
Zull: Autorizacion.





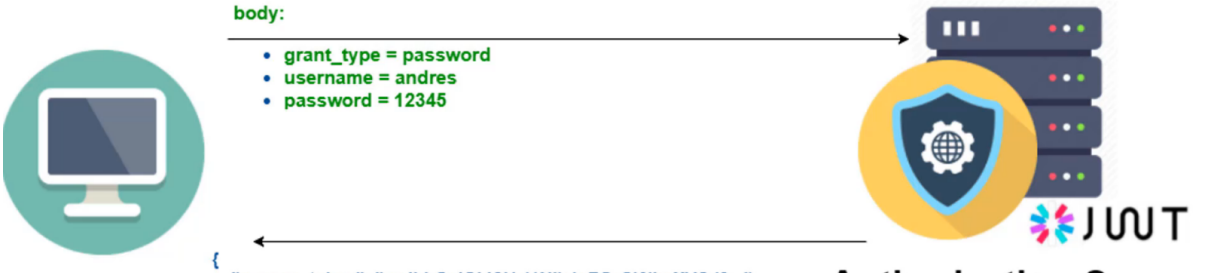
url: POST /auth/token

header:

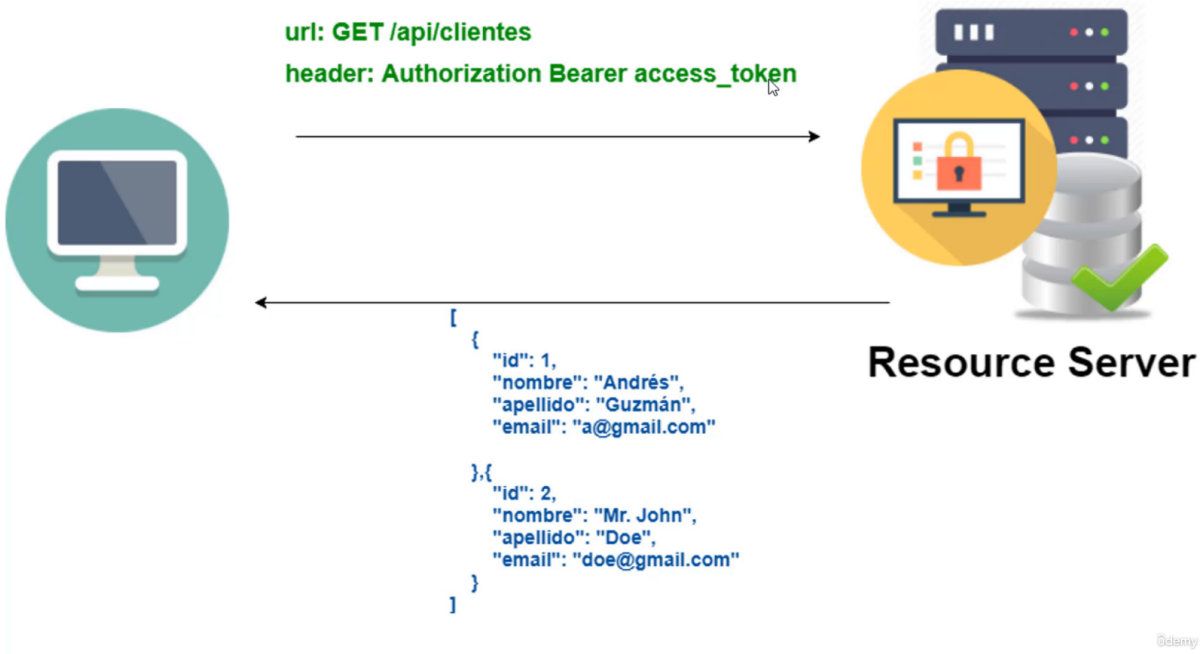
- Authorization: Basic Base64(client_id:client_secret)
- Content-Type: application/x-www-form-urlencoded

body:

- grant_type = password
- username = andres
- password = 12345

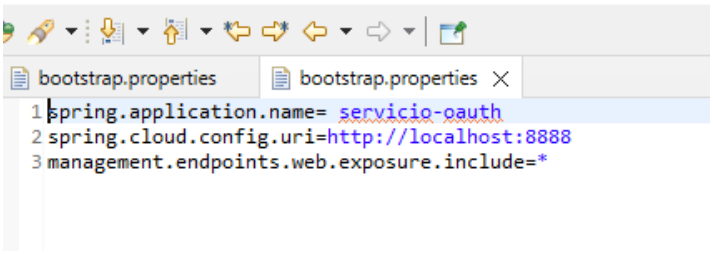



```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer",
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expires_in": 3599,
  "scope": "read write",
  "jti": "58efb674-46e6-4f6b-bbf0-e92e21e4b34a"
}
```

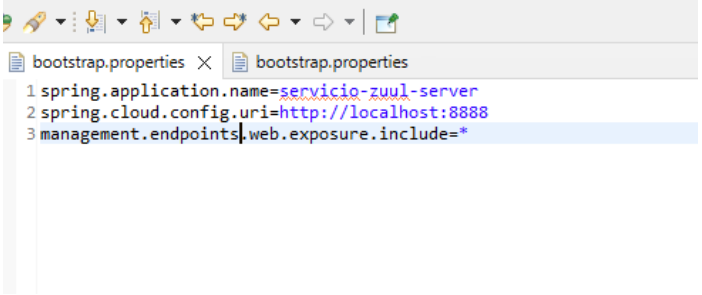


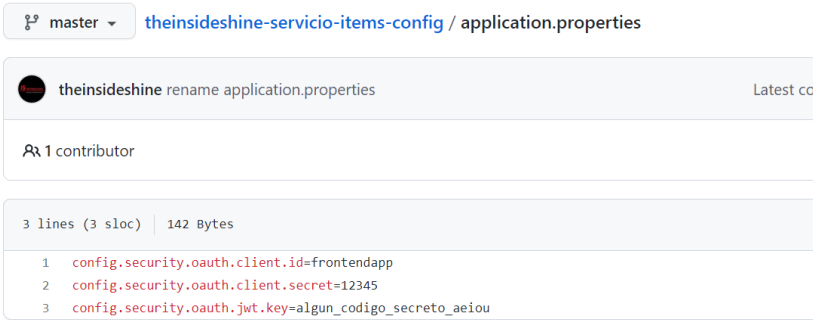
Cómo comparten las claves de jwt los el gateway Zuul y el Ms oauth, se configura el uso de el servidor de configuración en los dos.
para esto se usa el archivo bootstrap.properties

Suite 4



Tool Suite 4





MicroServicio Oauth: autenticacion

Debe incluirse las entidades alumnos, las dependencias para la comunicación entre microservicios y las de SpringSecurity

pom.xml
https://github.com/theinsideshine/microservicios-oauth/blob/main/springboot-servicio-oauth/pom.xml

application.properties
https://github.com/theinsideshine/microservicios-oauth/blob/main/springboot-servicio-oauth/src/main/resources/application.properties

appStart.java
https://github.com/theinsideshine/microservicios-oauth/blob/main/springboot-servicio-oauth/src/main/java/com/formacionbdi/microservicios/app/oauth/SpringbootServicioOauthApplication.java

Gateway zull: autorización

pom.xml

https://github.com/theinsideshine/microservicios-oauth/blob/main/springboot-servicio-zuul-server/pom.xml